REDUCING NUMBER OF CONSECUTIVE
ONES IN DATA DEPENDENT SCRAMBLER

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/472,172, filed on May 21, 2003, which is hereby incorporated by reference in its entirety.

FIELD OF THE INVENTION

[0002] The present invention relates to data coding in a communications channel, and more particularly to data coding that reduces the number of consecutive ones in a communications channel.

BACKGROUND OF THE INVENTION

[0003] Magnetic storage systems such as disk drives include a magnetic medium or platter with a magnetic coating that is divided into data tracks. The data tracks are divided into data sectors that store fixed-sized data blocks. A read/write head typically includes a write circuit and write element such as an inductor that selectively generates positive and negative magnetic fields that are stored by the magnetic medium. The stored positive and negative fields represent binary ones and zeros. The read/write head includes an element such as a magneto-resistive element that senses the stored magnetic field to read data from the magnetic medium. A spindle motor rotates the platter and an actuator arm positions the read/write head relative to the magnetic medium.

[0004]   Magnetic storage systems typically code the user data using Run Length Limited (RLL) code. RLL coding reduces sequences in the user data that may cause problems with timing circuits of the magnetic storage system. For example, RLL code may enforce constraints on the number of consecutive ones and/or zeros that are allowed to occur in the data. The efficiency of the RLL code is typically measured in terms of a code rate. For every m-bits or m-bytes of user data, an n-bit or n-byte encoded word is written to the storage media. RLL codes are used to eliminate unwanted bit patterns in the original data and typically do not have error correction capability. RLL coding, however, reduces data storage capacity by increasing channel bit density (CBD), which reduces a signal to noise ratio (SNR) and may lead to lower data reliability.

[0005]   Referring now to Figure 1, a write-path of a data storage system with RLL coding is shown. A host bus interface (HBI) 14 receives a user data sequence from a host computer 16. For example, the user data sequence may include 4096 bits of a sector, which are arranged in 410 10-bit symbols. A buffer manager (BM) 18 stores the user data sequence in a buffer 20 and then sends the user data sequence from the buffer 20 to a disk formatter (DF) 22 with proper timing. An ECC/CRC encoder 24 appends CRC and ECC bits to the user data sequence.

[0006]   The ECC bits are computed based on the user data sequence and the CRC bits. A scrambler 26 generates a pseudo-random sequence that is based on a polynomial and seed. The user data sequence and the scrambler sequence are then input to an XOR gate 27, which outputs a scrambled user

data sequence. A RLL encoder 28 is used to constrain the unwanted bit patterns in the scrambled sequence.

[0007]   To increase SNR and data storage capacity, data storage systems were developed without RLL coding using data-dependent scramblers (DDS). Data is processed by the DDS to eliminate unwanted bit patterns. The DDS is disclosed in "Improved Data Coding For Enforcing Constraints On Ones and Zeros In a Communications Channel", U.S. Patent Application Serial No. 10/423,552, filed April 25, 2003, which is commonly assigned and is hereby incorporated by reference in its entirety. The scrambled user data sequence from the DDS is forwarded to an ECC/CRC device, which generates and appends CRC and ECC bits to the scrambled user data.

[0008]   Because the CRC and ECC bits that are generated may also contain unwanted bit patterns, traditional RLL coding may be used to encode the ECC/CRC portion. The data storage system is still referred to as being without RLL coding because the CRC and/or ECC bits are relatively small in number as compared to the number of bits in the user data sequence.

[0009]   The data storage system using the DDS provides a global constraint (G) (i.e., longest length of a run of consecutive zeros) of $2*M$, where M is the symbol size (e.g., 10-bits). Such a data storage system further provides an interleave constraint (I) (i.e., longest run of consecutive zeros in an interleaved sub-sequence) of $2*(M-1)$. In the case of a 10-bit symbol size, the G/I constraint is 20/18. However, the maximum number of consecutive ones is $4*M-2$. In the case of a 10-bit symbol size, the maximum length of ones can be up to 38. It is desirable

to reduce the maximum number of consecutive ones while maintaining the G/I constraint.

SUMMARY OF THE INVENTION

[0010]   Accordingly, the present invention provides a data dependent scrambler (DDS) for a communications channel that transmits a user data sequence having a plurality of symbols. The DDS includes a scrambler that generates a scrambled user data sequence that is based on the user data sequence and a seed. A first encoder selectively interleaves adjacent symbols in the scrambled user data sequence if an all-zero symbol is produced by bit interleaving. The first encoder identifies a pivot bit that is adjacent to the all-zero symbol if interleaving is performed and replaces the all-zero symbol with an all-one symbol if the pivot bit is zero.

[0011]   In one feature, the DDS further includes a seed finder that selects the seed based on the symbols of the user data sequence.

[0012]   In another feature, the scrambler performs a bit-wise XOR operation on the user data sequence and the seed to generate the scrambled user data sequence.

[0013]   In another feature, the DDS further includes a code finder that selects an H-code based on symbols of the user data sequence. The DDS further includes a second encoder that reduces a Hamming weight of the scrambled user data sequence using the H-code, the seed and the scrambled user data sequence. The second encoder generates a first token and a second

token. The first token is determined based on the seed and the H-code. The first token is determined by a bit-wise XOR operation on the seed and the H-code. The second token is a ones complement of the first token.

[0014]   In still another feature, the second encoder includes a symbol lookup table, processes two adjacent symbols of the scrambled user data sequence at a time and selectively employs the first and second tokens and the symbol lookup table to encode the scrambled user data based on a Hamming weight of the two adjacent symbols.

[0015]   In yet another feature, the DDS is implemented in a write path of a data storage device.

[0016]   Further areas of applicability of the present invention will become apparent from the detailed description provided hereinafter. It should be understood that the detailed description and specific examples, while indicating the preferred embodiment of the invention, are intended for purposes of illustration only and are not intended to limit the scope of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017]   The present invention will become more fully understood from the detailed description and the accompanying drawings, wherein:

[0018]   Figure 1 is a functional block diagram of a data storage system with RLL coding according the prior art;

[0019]   Figures 2A and 2B are functional block diagrams of write and read paths, respectively, in a data storage system without RLL coding according ot the present invention;

[0020]   Figures 3A and 3B are functional block diagrams of encoding and decoding data dependent scramblers (DDS), respectively, of the data storage systems of Figures 2A and 2B, respectively;

[0021]   Figure 4 illustrates a user data sequence as it is processed through the DDS;

[0022]   Figures 5A, 5B and 5C illustrate adjacent symbols of the user data sequence before and after interleaving;

[0023]   Figure 6 is a flowchart illustrating user data processing through the DDS in the write path according to the present invention; and

[0024]   Figure 7 is a flowchart illustrating user data processing through the DDS in the read path according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0025]   The following description of the preferred embodiments is merely exemplary in nature and is in no way intended to limit the invention, its application, or uses. For purposes of clarity, the same reference numbers will be used in the drawings to identify the similar elements.

[0026]   While the present invention will be described in the context of a data storage system, skilled artisans will appreciate that the present invention can be applied to any binary communications channel with constraints on the

number of consecutive ones or zeros. As will be described further below, the data storage system according to the present invention does not employ RLL coding on a user data portion. The present invention discloses a coding technique that eliminates unwanted bit patterns with a smaller reduction in data storage capacity as compared to RLL coding. In other words, the coding technique according to the present invention reduces the channel bit density (CBD) less than data storage systems using RLL coding on the user data. As used herein, the term data dependent scrambler (DDS) is defined as a scrambler that alters at least one of a selected scrambler, a generating polynomial, a seed, and a scrambling sequence based upon current user data that is to be scrambled.

[0027]   Figure 2A illustrates a write-path of a data storage system without RLL coding according to the present invention. The host computer 16 transmits a user data sequence to the host bus interface (HBI) 14. The buffer manager (BM) 18 receives the user data sequence from the HBI 14 and controls the data flow through the write path. The user data sequence is stored in the buffer 20. The BM 18 passes the user data sequence on based on the timing of data flow through the write path. The disk formatter (DF) 22 receives the user data sequence from the BM 18 and formats the user data sequence based on the data storage device.

[0028]   A data dependent scrambler (DDS) encoder 40-E according to the present invention receives the user data sequence from the DF 22. The DDS encoder 40-E scrambles and processes the user data sequence to eliminate unwanted bit patterns. The DDS encoder 42-E appends a seed S and an H-code token (H) to the scrambled user data. Interleaving may also be performed. An

ECC/CRC encoder 42-E appends CRC bits that are calculated based on the DDS

encoded user data sequence. The ECC/CRC encoder 42-E also appends ECC

bits that are computed based on the DDS encoded user data sequence and CRC

bits. A RLL encoder 44-E encodes the appended ECC and CRC bits to reduce

unwanted bit patterns in the ECC/CRC bits.

[0029]   Referring now to FIG. 2B, a partial read path is shown. An RLL

decoder 44-D receives the read user data sequence, and decodes the RLL code to

recover the CRC and ECC bits and passes. The ECC/CRC decoder 42-D attempts

to correct all possible errors in the user data sequence. The DDS decoder 40-D

uses the seed S and the H-code token H to recover the user data sequence.

[0030]   Referring now to Figure 3A, the DDS encoder 40-E will be

described in further detail. The DDS encoder 40-E includes a data buffer 46 that

receives the user data sequence and coordinates data transfer timing through the

DDS 40-E. A seed/code finder 48 generates the seed (S) and the H-code token (H).

A first exclusive or (XOR) gate 50 performs a bit-wise XOR operation on the user

data sequence and the seed (S) to generate a scrambled user data sequence. An

H-code encoder 52 receives the scrambled user data sequence and encodes the

scrambled user data sequence to improve the worst-case Hamming weight thereof

using the H-code H and the seed S. A P-code encoder 54 receives the H-code

encoded user data sequence and selectively performs interleaving on the data

sequence to limit the maximum length of a run of zeros and ones in the sub-

sequences. A precoder 56 transfers all data in the interleaved nonreturn-to-zero-

invrse (INRZI) or nonreturn-to-zero-inverse (NRZI) domain into the nonreturn-to-zero (NRZ) domain.

[0031] In FIG. 3B, the P-code decoder 57 reverses the P-coding described above and below. The P-code decoder 57 outputs the H-code token H and the seed S to an H-code decoder 58, which reverses the H-code encoding that was performed above. The seed S is output to an XOR gate 59, which also receives the scrambled user data sequence. The XOR gate 59 performs bit-wise XOR on the scrambled user data and the seed to generate the user data sequence.

[0032] Referring now to Figures 3A and 4, the function of the DDS 40-E on the write path will be described with reference to changes in the user data sequence. An exemplary M-bit symbol data sequence ($D = \{D_{N-1}, D_{N-2}, ..., D_0\}$) of size N is used, where M = 10 (i.e., 10-bit symbol data sequence). For other symbol sizes or data lengths, similar DDS's can be constructed based on the principles of the present invention. Initially, the user data (D) is sent to the data buffer 46 and the seed/code finder 48. The seed (S) that is generated by the seed/code finder 48 is also input to the XOR gate 50. The delay of the data buffer 46 is sufficient to allow the scrambling sequence to be generated by the seed/code finder 48.

[0033] Given the exemplary data sequence D, there are 1023 non-zero 10-bit symbols. More specifically, if the number of bits in the symbol is M, it is always possible to find a seed S if $N < 2^M$. This is because not all of the 2^M different possible symbols can be included in a data sequence of fewer than 2^M symbols. Assuming that the user data includes 4096 bits organized into 10-bit symbols, there are at least 203 (1023 − 2 * 410) non-zero symbols that exist that are different from

any of the data symbols and their complements (i.e., bit-wise inversion). More particularly, the seed/code finder 48 searches symbols S and H in the set of {1, 2, 3, ..., 511} excluding the weight one symbols, such that S and H are different from any of the symbols and their complements in the user data symbols of set D. H, however, is preferably not the ones complement of S. Any one of the remaining symbols can be selected as S and H, which are sent to the first XOR gate 50 and H-code encoder 52.

[0034]   A scrambling sequence {S, S, ..., S} is formed by repeating seed S N times. The first XOR gate 50 performs bit-wise XOR of the scrambling sequence {S, S, ..., S} with the data sequence $D = \{D_{N-1}, D_{N-2}, ..., D_0\}$ to generate a scrambled sequence $C = \{C_{N-1}, C_{N-2}, ..., C_0\}$. The scrambled data sequence ($C = \{C_{N-1}, C_{N-2}, ..., C_0\}$) does not contain either an all-zero symbol or an all-one symbol. That is to say, every symbol in the data sequence is non-zero. At a minimum, each symbol may include only one "1" and nine "0"'s, providing a worst-case Hamming weight of 10% for the scrambled data sequence. In binary signaling, the Hamming weight is the number of "1" bits in the binary sequence.

[0035]   To improve the worst-case Hamming weight of the scrambled user data sequence C, the H-code encoder 52 is implemented. The H-code encoder 52 uses a token ($token_1$) and its complement ($token_2$) to indicate that coding has occurred. In this manner, the H-code decoder 58 on the read path can reverse the H-code encoding process. $Token_1$ is selected such that it is different than any of the scrambled user data symbols of set C and their complements and such that it is not the all "1" symbol. The H-code encoder 52 uses S XOR H as

token$_1$. Since H is different from any of the user data symbols and their complements, it follows that both token$_1$ and token$_2$ are different from any of the scrambled data symbols of the set C.

[0036]   A 10-bit to 6-bit look-up table maps the 10-bit symbols of weight one or weight two into 6-bit patterns having at least weight two. The 10-bit to 6-bit look-up table is a static table that can be stored in memory. There are ten weight one 10-bit symbols and forty-five weight two 10-bit symbols. In total, there are fifty-five 10-bit symbols to convert using the table. Further, there are fifteen weight two 6-bit patterns, twenty weight three 6-bit patterns, fifteen weight four 6-bit patterns, six weight five 6-bit patterns and one weight six 6-bit pattern. Therefore, the fifty-five weight one and weight two 10-bit symbols can be converted into fifty-five of the fifty-seven possible 6-bit symbols of weight two or greater. This enables a reversible correspondence, such as a one-to-one correspondence.

[0037]   The H-code encoder 52 processes the scrambled symbol set C two symbols at a time. If the total Hamming weight of the two symbol group is at least four, the data is passed on. If the total Hamming weight of the two symbol group is two or three (i.e., [1,1] or [1,2]), a symbol replacement occurs as follows: The H-code encoder 52 inserts token$_1$ for the symbol on the left (i.e., the first symbol of the two-symbol set). For the symbol on the right (i.e., the second symbol of the two-symbol set), the first four bits of the 10-bit symbol are used to indicate the position of the "1" in the weight one symbol. The second symbol is converted into a 6-bit symbol using the table described above. If the Hamming

weight of the two-symbol group is three with [2,1], the H-code encoder 52 inserts token$_2$ on the left. As described above, the H-code encoder 52 uses the first four bits of the 10-bit symbol to indicate the position of the "1" in the weight one symbol. The second symbol is converted into a 6-bit symbol using the table described above.

[0038]   A more detailed description of the operation and function of the H-code encoder 52 is provided in co-pending U.S. Patent Application Serial No. 10/693,796, entitled "Methods and Apparatus for Improving Minimum Hamming Weights of a Sequence", filed on August 12, 2003, the disclosure of which is hereby incorporated by reference in its entirety.

[0039]   The H-code data sequence does not contain either the all-zero symbol or the all-one symbol. As a result, a global constraint (G) is 18. This means that there are at most, 18 consecutive zeros in the H-coded data sequence. However, it is still possible to have a long run of zeros in the sub-sequences. The P-code encoder 54 according to the present invention limits the maximum length of a run of zeros in the sub-sequences by selectively interleaving adjacent symbols as will be described below.

[0040]   Referring now to Figures 5A, 5B, 5C and 6, the P-code coding process will be described in detail. In step 500, an index (j) is set equal to 1 (j = 1). The P-code encoder 54 processes the scrambled data sequence (C) two adjacent symbols at a time. If an all-zero symbol occurs after bit-interleaving as determined in step 504, bit interleaving is performed in step 506. Otherwise,

control determines whether there are additional symbols in step 508. If true, j is set equal to j+2 in step 510. Otherwise, control ends.

[0041]   After bit interleaving is performed in step 506, an all-zero symbol is generated and the other non-zero symbol has a Hamming weight of at least four. Bit interleaving of the two symbols may generate symbols that are the same as either $token_1$ or $token_2$. This, however, does not create any difficulties when decoding because a P-code decoder 57 processes the data before the H-code decoder 58. Further, the P-code decoded data sequence does not contain any tokens other than those generated by the H-code encoder 52.

[0042]   In step 511, a pivot bit is identified. The pivot bit is the bit of the non-all-zero symbol that is adjacent to the all-zero symbol as shown in the examples in FIGs. 5B and 5C. The P-code encoder 54 determines whether the pivot bit is zero in step 512. If the pivot bit is zero, the P-code encoder replaces the all-zero symbol by the all-one symbol in step 514. If the pivot bit is one, the P-code encoder continues with step 508 as described above.

[0043]   Referring now to FIG. 7, when decoding the P-coded data sequence, the P-code decoder 57 processes the symbols of the data sequence two at a time. In step 550, control sets j=1 and reads adjacent symbols in step 552. If there is an all-zero symbol in the two symbol group as determined in step 554, bit interleaving is performed in step 556. If there is an all-one symbol and a non-all-zero symbol in a group as determined in step 558, the all-one symbol is replaced by the all-zero symbol in step 560 and bit interleaving is performed in step 562.

[0044]   As a result of the P-code encoding, the global constraint (G) is increased. The longest length of a run of zeros is 2*M, where M is the symbol size. The interleave constraint (I) is 2*(M-1). In the case of a 10-bit symbol, the DDS achieves a G/I constraint of 20/18 and a minimum Hamming weight of 20%. Further, the maximum length of a run of ones is reduced to 20, as compared to 38 in prior systems with M=10.

[0045]   Those skilled in the art can now appreciate from the foregoing description that the broad teachings of the present invention can be implemented in a variety of forms. Therefore, while this invention has been described in connection with particular examples thereof, the true scope of the invention should not be so limited since other modifications will become apparent to the skilled practitioner upon a study of the drawings, the specification and the following claims.